

Upgrade



# How to upload your python package to PyPi



joelbarmettlerUZH   Follow

May 7 · 8 min read

The PyPi package index is one of the properties that makes python so powerfull: With just a simple command, you get access to thousands of cool libraries, ready for you to use. In this short introduction, I am going to explain how you can upload upload your own code to PyPi and let others profit from your inventions. On the following pages, I am going to show you the following steps:

- Make your code publish-ready
- Create a python package
- Create the files PyPi needs
- Create a PyPi account
- Upload your package to github.com
- Upload your package to PyPi
- Install your own package using pip
- Change your package



Complete Guide of how to upload your package to PyPi and install it via pip. Screenshot: PyPi.org

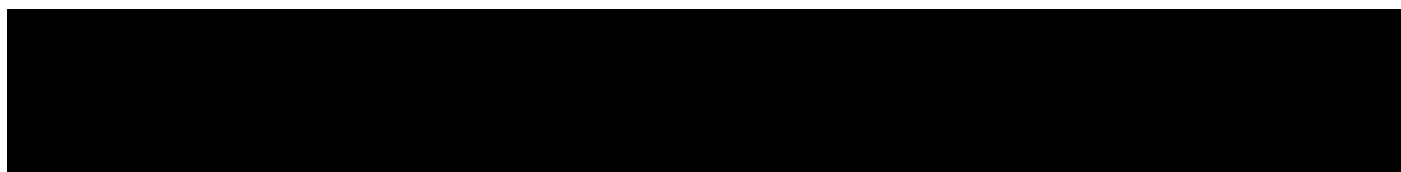
## But what is PyPi?

Well, according to their Website:

The Python Package Index (PyPI) is a repository of software for the Python programming language. PyPI helps you find and install software developed and shared by the Python community. Learn about installing packages. Package authors use PyPI to distribute their software. Learn how to package your Python code for PyPI.

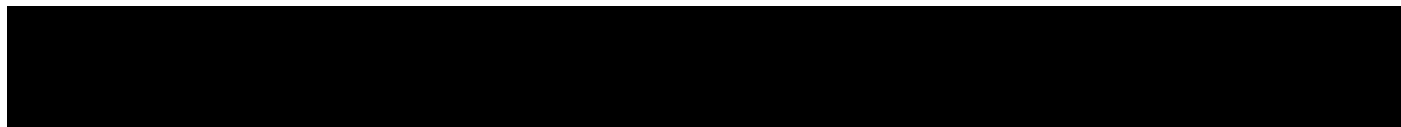
## How to use PyPi

I am sure most of you have already installed PyPi (*pip*) and worked with it. If you have not, you definitely should! It's as simple as downloading this file. Now, open your command prompt and navigate via "cd" into the directory where the downloaded file is located. Then, run the command "python get-pip.py".

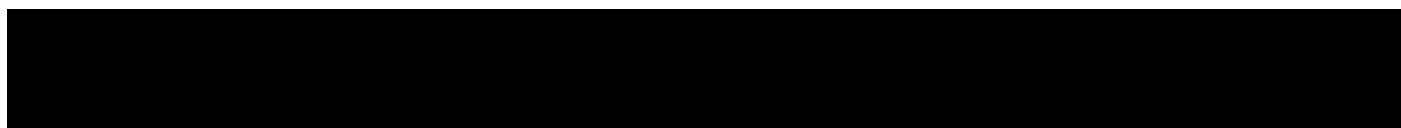


To use pip, type "pip insall *packagename*" into the console. When you want to install

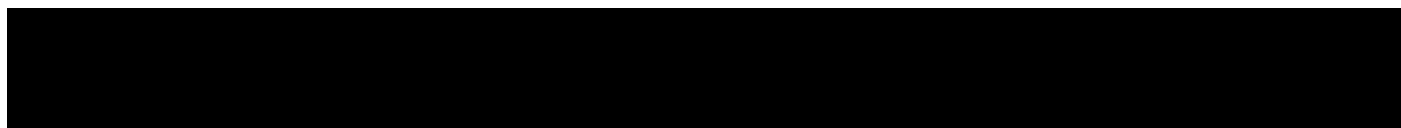
*scrapeasy*, you type the following command:



Did you get an error that says **‘pip’ is not recognized as an internal or external command, operable program or batch file.**? Don’t panic, we got this. Your pip script is not in the windows environment variables, so you can not access it from everywhere on your system. Let’s fix this with the following command:



Or, as an example



It should work now.

## Make your code publish-ready

Let’s prepare your code for the uploading. First, you should remove all “print” statements from your code. It’s annoying when you work with a library and your command prompt is flooded with print messages that are not yours—therefore remove all of them. If you want to inform the user about certain activities, use logging.

Also make sure to not include code that exists outside of a class or a function, otherwise

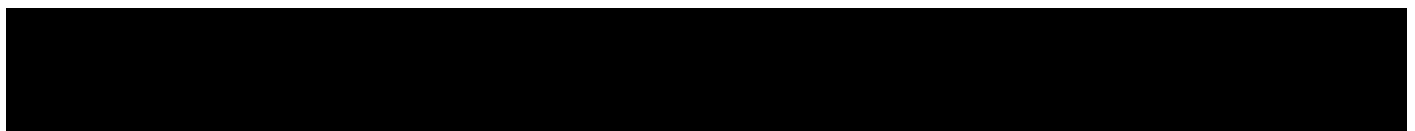
this code will run every time somebody imports your library. If you want to include example code into the classes (which is legit), wrap it into the “\_\_main\_\_” function.



## Create a python package

To create a package, create a folder that is named exactly how you want your package to be named. Place all the files and classes that you want to ship into this folder.

Now, create a file called `__init__.py` (again two underscores). Open the file with your text editor of choice. In this file, you write nothing but import statements that have the following schema:



The `__init__.py` file is used to mark which classes you want the user to access through the package interface. Let's make an example. Assume you want to upload a library named “MyLib” to PyPi. First, create a Folder called “MyLib” and place your classes inside of it.

MyLib

-Class1.py

-Class2.py

Then, remove all print-statements and all the code that does not sit inside of a class.

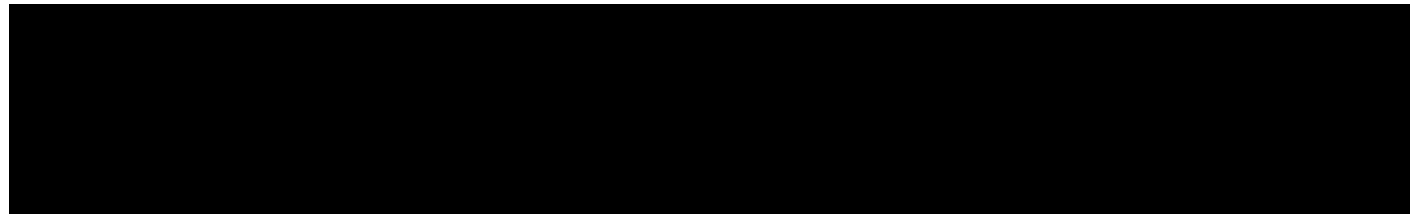
Finally, create the `__init__.py`-file and import all the methods that the user should access.

MyLib

- `__init__.py`

-File1.py

-File1.py



## Create a PyPi account

You can register yourself for a PyPi account [here](#). Remember your username (not the Name, not the E-Mail Address) and your password, you will need it later for the upload process.

## Upload your package to github.com

Create a github repo including all the files we are going to create in a second, as well as your package folder. Name the repo exactly as the package. You find an example of such a repo [here](#).

## Create the files PyPi needs

PyPi needs three files in order to work:

- `setup.py`
- `setup.cfg`
- `LICENSE.txt`
- `README.md` (*optinal but highly recommended*)

Place all these files outside of your package folder:

MyLib

`setup.py`

`setup.cfg`

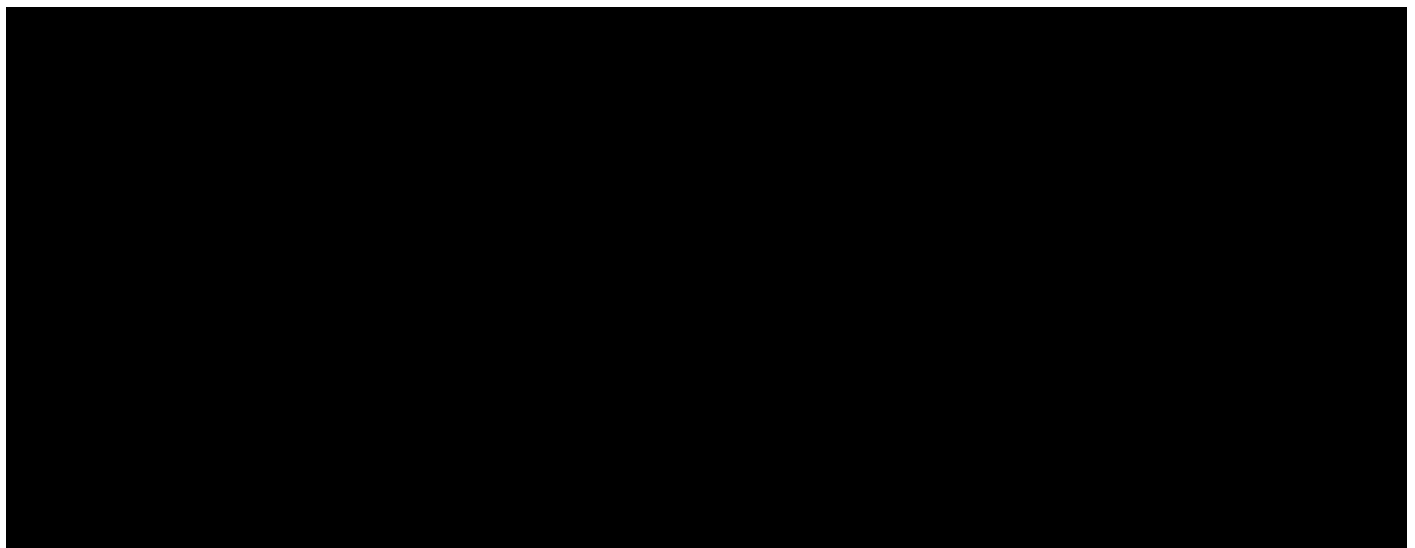
`LICENSE.txt`

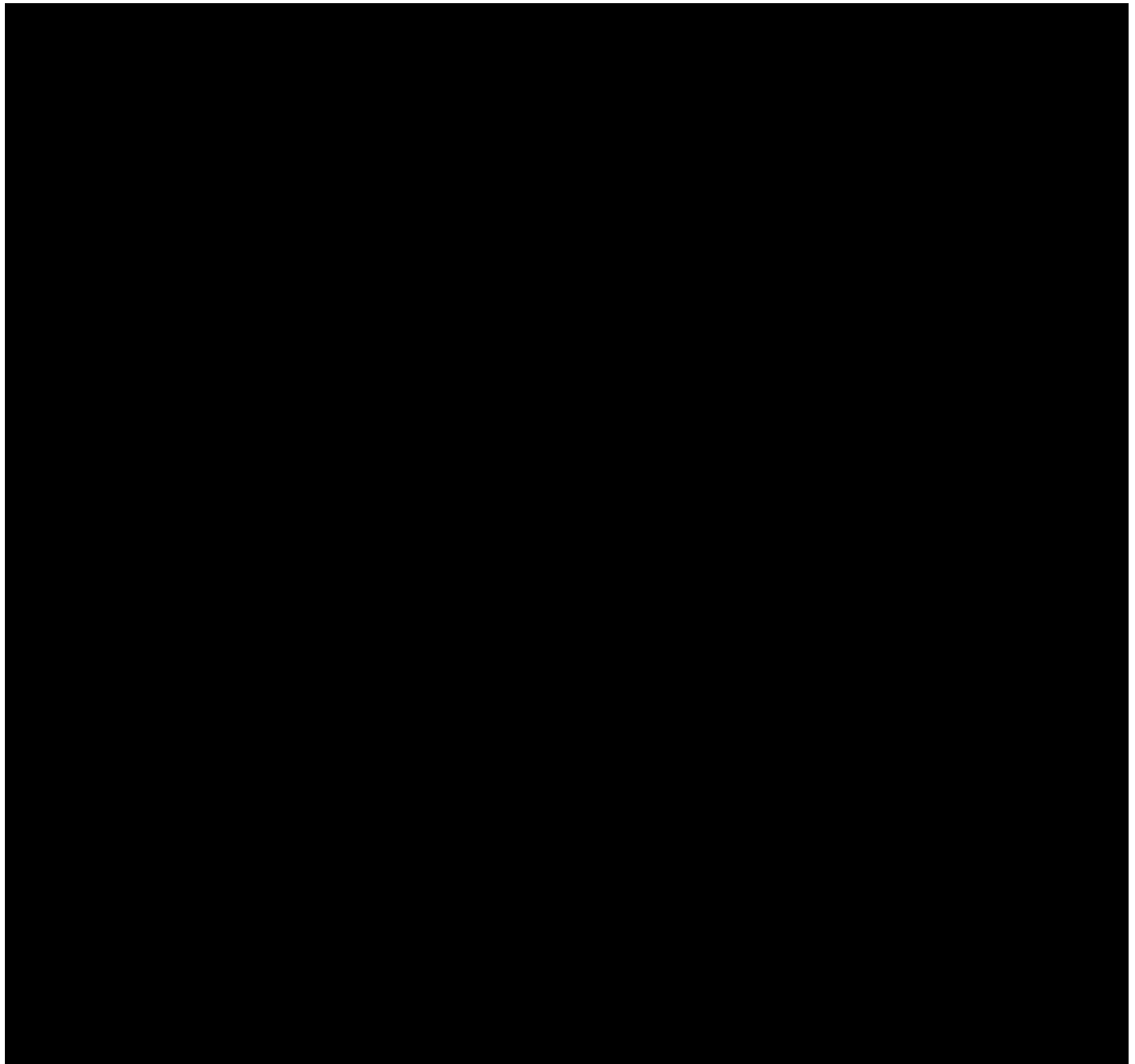
`README.md`

Let's go through this files one-by-one

## `setup.py`

The `setup.py` file contains information about your package that PyPi needs, like its name, a description, the current version etc. Copy and Paste the following Code and replace the Strings with your matching content:





Let me explain the `download_url` and the `install_requires` a bit further.

## `download_url`

You have previously uploaded your project to your github repository. Now, we create a new release version of your project on github. This release will then be downloaded by anyone that runs the “`pip install YourPackage`” command.

First, go to [github.com](https://github.com) and navigate to your repository. Next, click on the tab “releases”

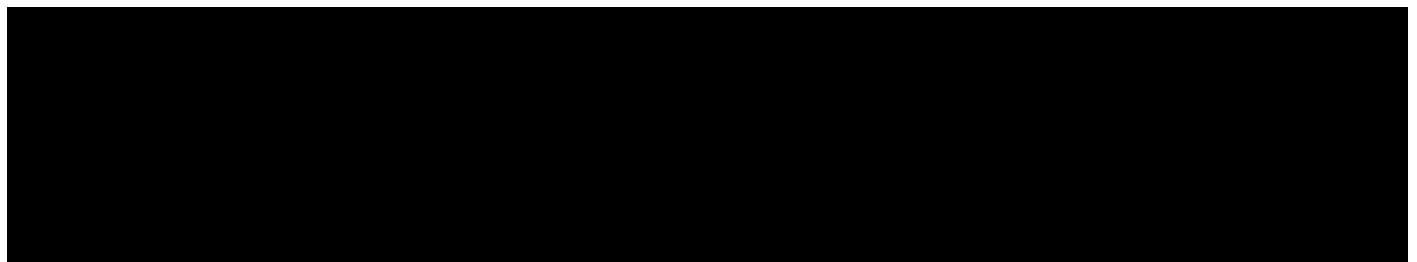


and then on “Create a new release”. Now, define a Tag version (it is best to use the same number as you used in your setup.py version-field: `v_01`). Add a release title and a description (not that important), then click on “publish release”. Now you see a new release and under **Assets**, there is a link to **Source Code (tar.gz)**. Right-click on this link and chose *Copy Link Address*. Paste this link-address into the `download_url` field in the setup.py file. Every time you want to update your package later on, upload a new version to github, create a new release as we just discussed, specify a new release tag and copy-paste the link to Source into the setup.py file (do not forget to also increment the version number).

## Install\_requires

Here, you define all the dependencies your package has—all the pip packages that you are importing. Simply speaking: Go through all your Classes and watch what you have imported. Every package that you have imported and downloaded via pip (most propably all, except the standard libraries) must be listed here as an `install_requires` by adding the package name.

Let me give you an example



The packages “numpy” and “scrapeasy” were downloaded via pip, so you need to add “numpy” and “scrapeasy” in your dependencies. But how do you know whether a package is a standard library or downloaded via pip? Well, simply open your cmd and type *pip install packagename*. If you get “**Requirement already satisfied**”, you are good to go. If you get an error stating “**Could not find a version that satisfies the**

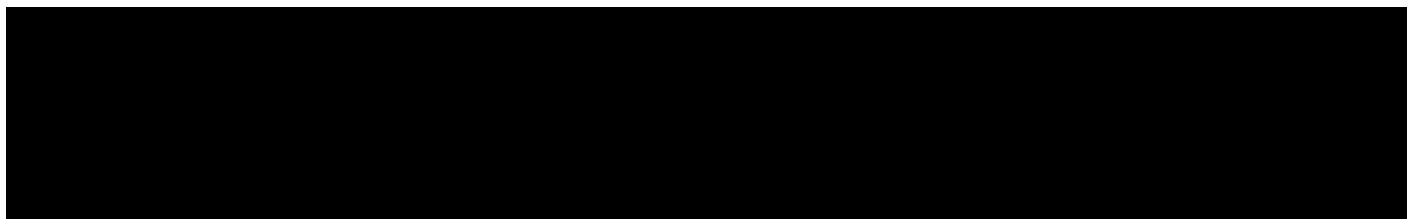
**requirement”,** you know that this package is either not installed via pip or a standard library.

The resulting *install\_requires* field would look like this:



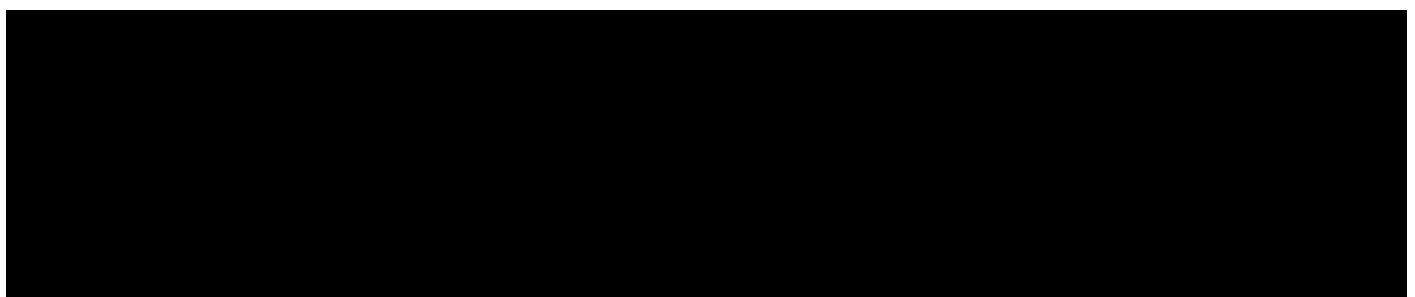
## setup.cfg

This is a short one. Create a new file called “setup.cfg”. If you have a description file (and you definitely should!), you can specify it here:



## LICENSE.txt

Use this file to define all license details. Most probably, just copy-paste the license text from this website into the LICENSE.txt file. I always use the MIT license for my projects, but feel free to use your own.



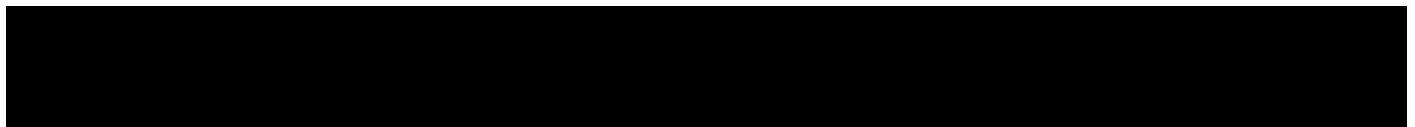


## README.md (Optional)

Create a markdown file with [this beautiful website](#), then download it and place it into your directory.

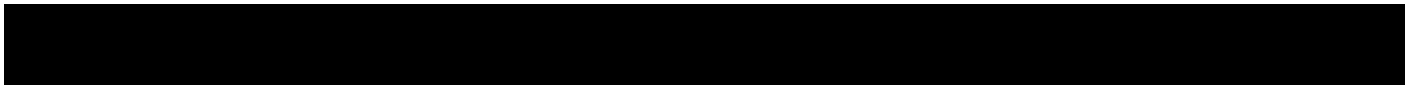
## Upload your package to PyPi

Now, the final step has come: uploading your project to PyPi. First, open the command prompt and navigate into your the folder where you have all your files and your package located:



Now, we create a source distribution with the following command:






You might get a warning stating “Unknown distribution option: ‘install\_requires’”. Just ignore it.

We will need *twine* for the upload process, so first install twine via pip:



Then, run the following command:



You will be asked to provide your username and password. Provide the credentials you used to register to PyPi earlier.

**Congratulations, your package is now uploaded!** Visit

<https://pypi.org/project/YOURPACKAGENAME/> to see your package online!

## Install your own package using pip

Okay, now let's test this out. Open your console and type the following command:

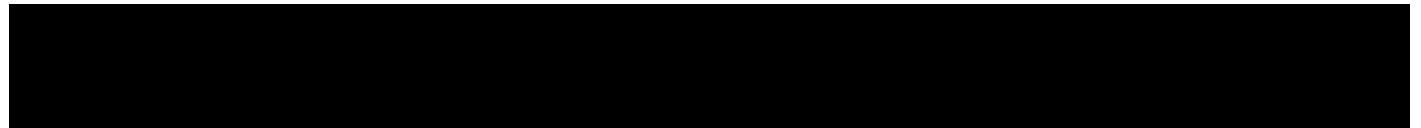
It works! Now open the python SHELL/IDLE and import your package.

## Change your package

If you maintain your package well, you will need to change the source code form time to time. This is easy. Simply upload your new code to github, create a new release, then adapt the setup.py file (new download\_url—according to your new release tag, new version), then run the setup.py and the twin command again (navigate to your folder first!)



Finally, update your package via pip to see whether your changes worked:



Some rights reserved



46 claps



joelbarmettlerUZH

Follow

Student @ University of Zurich UZH Major: Software Systems Minor: Neuroinformatics



Also tagged Programming

...



9 min read

2.3K |



Also tagged Programming

...



25 min read

1.8K |



Also tagged Python

...



9 min read

2.8K |

Responses



Be the first to write a response...